7. XML – DTD - XSD

www.LearnDB.com

Dr. Imed Bouchrika
Dept of Mathematics & Computer Science
University of Souk-Ahras
imed@imed.ws

Part I

XML

XML is

- XML : e**X**tended **M**arkup **L**anguage
- XML is a format for
  - Representing data in a structured way.
  - Storing Data
  - Transferring & Communicating Data.
- You invent the structure you want with XML

XML : why

- XML is a widely accepted open standard.
- XML allows to clearly separate content from form (appearance)
- XML is text-oriented.
- XML is extensible.
- XML is self-describing.
- XML is universal; meaning internationalization is no problem.

XML : why

- XML is independent from platforms and programming languages.
- XML provides a robust and durable format for information storage.
- XML is easily transformable and transferrable.
- XML is a future-oriented technology.

XML : contains

- The first line is the XML declaration.
  <?xml version="1.0" encoding="utf-8">

- Root Element :
  <?xml version="1.0" encoding="utf-8">
  <library>
  .....
  </library>

- Elements : **<anything-you-want>** ….

- Attributes : <anything-you-want **any-name="any-value"**>

XML : how

- Text-oriented ?
  - ➔ Open any text editor , New file (extension preferably .xml)
    - Notpad, WordPad, Eclipse, Notepad++ ....
- Write your XML code using Tags

- Tag ? **<anything-you-want>** **..**Content**.. </anything-you-want>**

- You can even view your XML files using your Web Browser
  - Firefox, chrome…..

- But XML documents needs to be **Well-Formed** and **Valid !**

Well-Formed [1]

- There must be only one ROOT

```
<?xml version="1.0" ?>
<library>
  <book>Java</book>
</library>
<shop>
  <product>Juice</product>
</shop>
```

```
<?xml version="1.0" ?>
<city>
 <library>
   <book>Java</book>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

Well-Formed [2]

- Every Element Must Have a Closing Tag

```
<?xml version="1.0" ?>
<city>
 <library>
   <book>Java
   <active>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

```
<?xml version="1.0" ?>
<city>
 <library>
   <book>Java</book>
   <active/>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

Well-Formed [3]

- XML Tags are Case Sensitive

```
<?xml version="1.0" ?>
<city>
 <library>
   <Book>Java<book>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

```
<?xml version="1.0" ?>
<city>
 <library>
   <BoOk>Java<BoOk>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

Well-Formed [4]

- XML Tags must not overlap

```
<?xml version="1.0" ?>
<city>
 <library>
   <book><name>Java<book></name>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

```
<?xml version="1.0" ?>
<city>
 <library>
   <book><name>Java</name><book>
 </library>
 <shop>
   <product>Juice</product>
 </shop>
</city>
```

Well-Formed [5]

- Attribute values must be enclosed by quotes.

```
<?xml version="1.0" ?>
<city>
 <library>
   <book lang=english>Java<book>
 </library>
 <shop>
   <product price=19>Juice</product>
 </shop>
</city>
```

```
<?xml version="1.0" ?>
<city>
 <library>
   <book lang="english">Java<book>
 </library>
 <shop>
   <product price="19">Juice</product>
 </shop>
</city>
```

Well-Formed [5]

- XML documents must start with a declaration line.

`<city>`
`  <library>`
`  <book lang=english>Java<book>`
`  </library>`
`  <shop>`
`  <product price=19>Juice</product>`
`  </shop>`
`</city>`
	

`<?xml version="1.0" ?>`
`<city>`
`  <library>`
`  <book lang="english">Java<book>`
`  </library>`
`  <shop>`
`  <product price="19">Juice</product>`
`  </shop>`
`</city>`
	

Part II

DTD

DTD : what

- DTD = Document Type Definition
- DTD defines **a set of rules** describing The **structure of an XML** document with a list of legal elements and attributes.
- XML Document is **VALID** if it **complies with** or **respect** such set of rules defined with the DTD.
- DTD has its own syntax.

DTD : structure

- DTD structure is as follows:

```
<!DOCTYPE ROOT_ELEMENT [
     .....
     .....
]>
```

- DTD has three basic components :
  - **!ELEMENT**
  - **!ATTLIST**
  - **!ENTITY**

DTD Syntax: Empty Element

- Empty elements are declared with the category keyword EMPTY:

```
<!ELEMENT element-name EMPTY>
```

```
<!ELEMENT br EMPTY>
```

- In XML :

```
<br/>
```

DTD Syntax: Element with Data

- Elements with only **P**arsed **C**haracter **DATA** are declared with #PCDATA inside parentheses:

```
<!ELEMENT element-name (#PCDATA)>
```

```
<!ELEMENT title (#PCDATA)>
```

- XML

```
<title>Java Programming</title>
```

DTD Syntax: Element with Sub-Elements

- Elements with one or more children are declared with the name of the children elements inside parentheses

```
<!DOCTYPE book [
   <!ELEMENT book (title, author)>
   <!ELEMENT title (#PCDATA)>
   <!ELEMENT author (#PCDATA)>
   ]
   >
```

- XML :

```
<book>
   <title>Feature Extraction</title>
   <author>Mark Nixon</author>
   </book>
```

DTD Syntax: Occurrences : 1 * + ?

- Default is 1
- ? : Zero or One
- * : Zero or more
- + : One or more

```
<!ELEMENT element-name (child+)>
```

```
<!ELEMENT parent (wife?, son*,daughter*)>
```

```
<!ELEMENT parent (wife?, (son,daughter)* )>
```

```
<!ELEMENT parent (wife?, son*,daughter*, house+)>
```

DTD Syntax: Choice for Elements

- An element can have a choice for Sub-elements using |

```
<!ELEMENT element-name (option1|option2)>
```

```
<!ELEMENT student ( pass | fail ) >
```

- XML

```
<student>
   <fail>9.10</fail>
<student>
```

- Or

```
<student>
   <pass>9.10</pass>
<student>
```

DTD Syntax: Attributes

- To declare an attribute, we use **<!ATTLIST>**

```
<!ATTLIST element-name attr-name attr-type attr-value>
```

- DTD

```
<!ATTLIST book language CDATA "english">
```

```
<!ATTLIST country code CDATA #REQUIRED>
```

```
<!ATTLIST country code CDATA #IMPLIED>
```

```
<!ATTLIST car wheels CDATA #FIXED "4 ">
```

- XML

```
<book language="french">Java</book>
```

DTD Syntax: Choice for Attributes

- Choice for attribute values is done using the |

```
<!ATTLIST Bus color (white|blue|green) "white">
```

DTD Syntax: Entities

- Entities are variables used to define shortcuts to standard text or special characters.

```
<!ENTITY mycopyright "Copyright by Imed Bouchrika">
```

- XML

```
<mywebsite> &copyright; <mywebsite>
```

DTD Integration: Standalone="yes"

- DTD can be integrated within an XML document using the attribute **standalone="yes"** within the declaration line:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE parent [
   <!ELEMENT parent (son,daughter)* >
   <!ATTLIST parent gender (male|female) "male">
   <!ATTLIST parent name CDATA #REQUIRED>
   <!ELEMENT son (#PCDATA) >
   <!ELEMENT daughter (#PCDATA) >
]>
<parent gender="male" name="Imed">
<son>Hou</son>
<daughter>Asma</daughter>
</parent>
```

DTD Integration: Standalone="no"

- Save your DTD to file called for example : **mydata.dtd**

```
<!DOCTYPE parent [
<!ELEMENT parent (son,daughter)* >
<!ATTLIST parent gender (male|female) "male">
<!ATTLIST parent name CDATA #REQUIRED>
<!ELEMENT son (#PCDATA) >
<!ELEMENT daughter (#PCDATA) >
]>
```

- Reference your DTD file from the XML document

```
<?xml version="1.0" standalone="no"?>
    <!DOCTYPE note SYSTEM "mydata.dtd">
    <parent gender="male" name="Imed">
    <son>Hou</son>
    <daughter>Asma</daughter>
    </parent>
```

DTD Integration: Standalone="no"

- Save your DTD to file called for example : **mydata.dtd**

```
<!DOCTYPE parent [
   <!ELEMENT parent (son,daughter)* >
   <!ATTLIST parent gender (male|female) "male">
   <!ATTLIST parent name CDATA #REQUIRED>
   <!ELEMENT son (#PCDATA) >
   <!ELEMENT daughter (#PCDATA) >
]>
```

- Reference your DTD file from the XML document

```
<?xml version="1.0" standalone="no"?>
    <!DOCTYPE note SYSTEM "http://www.web.cc/mydata.dtd">
    <parent gender="male" name="Imed">
    <son>Hou</son>
    <daughter>Asma</daughter>
    </parent>
```

Part III

XSD

XSD : what

- XSD = XML Schema
- XSD defines **a set of rules** describing The **structure of an XML** document with a list of legal elements and attributes.
- XML Document is **VALID** if it **complies with** or **respect** such set of rules defined with the XSD.
- XSD is written in XML.

XSD : why

- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML
- XML Schemas support data types
- XML Schemas support namespaces

XSD : structure

- XSD structure is as follows:

```
<?xml version="1.0"?>
<xs:schema>
...
...
</xs:schema>
```

- XSD has a number of basic components :
  - **<xs:element>**
  - **<xs:attribute>**
  - **<xs:complexType>**
  - **<xs:sequence>**
  - **<xs:choice> ....**

XSD : Example : XML

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XSD : Example : DTD

```
<!DOCTYPE note[
      <!ELEMENT note (to, from, heading, body)>
      <!ELEMENT to (#PCDATA)>
      <!ELEMENT from (#PCDATA)>
      <!ELEMENT heading (#PCDATA)>
      <!ELEMENT body (#PCDATA)>
]>
```

XSD : Example : XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com" elementFormDefault="qualified">
    <xs:element name="note">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="to" type="xs:string"/>
                <xs:element name="from" type="xs:string"/>
                <xs:element name="heading" type="xs:string"/>
                <xs:element name="body" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XSD Syntax: Empty Element

- Empty elements are written as normal XML empty elements :

```
<xs:element name="br" />
```

```
<xs:element name="br" ></xs:element>
```

- XML

```
<br />
```

XSD Syntax: Element

- Elements with **only content inside** are declared with their corresponding data type:

```
<xs:element name="element-name" type="data-type">
```

```
<xs:element name="title" type="xs:string"></xs:element>
```

- Data Types:

  xs:string , xs:decimal , xs:integer , xs:boolean , xs:date , xs:time

- XML

```
<title>Java Programming</title>
```

XSD Syntax: Element : Default/Fixed data

- Element data can have a default or fixed value inside :

```
<xs:element name="element-name" type="data-type" default="value">
```

```
<xs:element name="element-name" type="data-type" fixed="value">
```

```
<xs:element name="city" type="xs:string" default="York" />
```

- XML

```
<city>York</city>
```

XSD Syntax: Sub Elements

- We use the ***xs:complexType*** with ***xs:sequence***

```
<xs:element name="book">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="title" type="xs:string"/>
         <xs:element name="author" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

- XML

```
<book>
   <title>Feature Extraction</title>
   <author>Mark Nixon</author>
</book>
```

XSD Syntax: Sub Elements

- **Would it work if changing the order of the sequence ?**

```
<xs:element name="book">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="title" type="xs:string"/>
         <xs:element name="author" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

- XML

```
<book>
   <author>Mark Nixon</author>
   <title>Feature Extraction</title>
</book>
```

XSD Syntax: Occurrences

- Occurrence is specified using : **maxOccurs** and **minOccurs**.
- For unlimited, we use the keyword: **unbounded**
- The default value for maxOccurs and minOccurs is **1**

XSD Syntax: Occurrences

- XSD

```
<xs:element name="husband">
  <xs:complexType>
    <xs:sequence>
       <xs:element name="wife" type="xs:string" maxOccurs="1"/>
       <xs:element name="son" type="xs:string" maxOccurs="unbounded" />
       <xs:element name="daughter" type="xs:string" minOccurs="0" />
       <xs:element name="parent" type="xs:string" minOccurs="2"
                                                  maxOccurs="2" />
    </xs:sequence>
  </xs:complexType>
 </xs:element>
```

- XML

```
<husband>
    <wife>Jane</wife>
    <son>Tim</son>
    <son>Rich</son>
    <parent>John</parent>
    <parent>Marry</parent>
</husband>
```

XSD Syntax: Choice

- XSD

```
<xs:element name="Doctor">
    <xs:complexType>
      <xs:choice>
        <xs:element name="Hospital" type="xs:string" />
        <xs:element name="Clinic" type="xs:string"/>
      </xs:choice>
    </xs:complexType>
</xs:element>
```

- XML

```
<Doctor>
    <Clinic>SouthWest Clinic</Clinic>
    </Doctor>
```

XSD Syntax: Attributes: Element no data

- That's an attribute of an element **without text data inside**.

```
<xs:element name="Book">
 <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="lang" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

- XML

```
<book lang="en">
    <title>Feature Extraction</title>
</book>
```

DTD Syntax: Attributes: Element with data

- XSD

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title">
       <xs:complexType>
         <xs:simpleContent>
           <xs:extension base="xs:string">
             <xs:attribute name="code" type="xs:string" />
           </xs:extension>
         </xs:simpleContent>
       </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="lang" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

- XML

```
<book lang="en">
    <title code="AB">Feature Extraction</title>
</book>
```

XSD Syntax: Attributes: Element with data

- XSD

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title">
        <xs:complexType mixed="true">
          <xs:attribute name="code" type="xs:string" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="lang" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

- XML

```
<book lang="en">
    <title code="AB">Feature Extraction</title>
</book>
```

XSD Syntax: Pointers

```
<xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title" />
       </xs:sequence>
       <xs:attribute name="lang" type="xs:string"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="dvd">
      <xs:complexType>
         <xs:sequence>
           <xs:element ref="title" />
         </xs:sequence>
       </xs:complexType>
    </xs:element>
</xs:element>

<xs:element name="title" >
  <xs:complexType mixed="true">
      <xs:attribute name="code" type="xs:string" />
   </xs:complexType>
</xs:element>
```

```
<book lang="en">
 <title code="AB">Feature Extraction</title>
</book>
<dvd>
  <title code="AB">Need for Speed 7</title>
</dvd>
```

XSD Syntax: Restriction

```
<xs:element name="weight">
    <xs:complexType>
        <xs:simpleContent>
            <xs:restriction base="xs:integer">
                <xs:minInclusive value="60"/>
                <xs:maxInclusive value="80"/>
                <xs:attribute name="unity" type="xs:string" />
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

```
<weight unity="kg">72</weight>
```

XSD Syntax: Choice of Element Data

```
<xs:element name="car">
   <xs:complexType>
      <xs:simpleContent>
        <xs:restriction base="xs:string">
           <xs:enumeration value="Chevy"/>
           <xs:enumeration value="Lexus"/>
           <xs:enumeration value="Audi"/>
           <xs:attribute name="price" type="xs:integer" />
        </xs:restriction>
   </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

```
<car price="80000">Audi</car>
```

XSD Syntax: Pattern : RegEx

```
 <xs:element name="name">
    <xs:complexType>
      <xs:simpleContent>
        <xs:restriction base="xs:string">
          <xs:pattern value="[a-z]{4}"/>
          <xs:attribute name="age" type="xs:integer" />
       </xs:restriction>
     </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

```
<name age="45">mark</car>
```

XSD Syntax: Creating Data Types

```
 <xs:element name="myp" type="persontype"/>
    <xs:complexType name="persontype">
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
 <myp>
    <firstname>Asma</firstname>
    <lastname>Bouchrika<lastname>
</myp>
```

XSD Syntax: Inheritance

```
<xs:element name="student" type="studenttype"/>

<xs:complexType name="persontype">
   <xs:sequence>
     <xs:element name="firstname" type="xs:string"/>
     <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="studenttype">
  <xs:complexContent>
    <xs:extension base="persontype">
       <xs:sequence>
         <xs:element name="university" type="xs:string"/>
       </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XSD Syntax: Inheritance

```
 <myp>
    <firstname>Asma</firstname>
    <lastname>Bouchrika<lastname>
</myp>
```

```
<student>
    <firstname>Mark</firstname>
    <lastname>Nixon<lastname>
    <university>Uni of Reading</university>
</student>
```

XSD Integration:

- XSD document are usually saved on a separate file with extension (xsd). An example is shown below for a file saved as : family.xsd

```
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="parent">
     <xs:complexType>
       <xs:sequence>
         <xs:element type="xs:string" name="son"/>
         <xs:element type="xs:string" name="daughter"/>
        </xs:sequence>
       </xs:complexType>
    </xs:element>
</xs:schema>
```

XSD Integration:

- **http://www.w3schools.com is for the namespace.**

- **Family.xsd is the filename for XSD file**

```
<?xml version="1.0"?>
    <parent
    xmlns="http://www.w3schools.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3schools.com family.xsd">
    <son>Hou</son>
    <daughter>Asma</daughter>
</parent>
```

DTD vs. XSD

DTD	XSD
DTD has its own Syntax : Additional Parser Required	XSD is written in XML. Only one XML parser is needed.
Basic Data Types only (#PCDATA, CDATA,..)	Richer Data Types (xs:string , xs:decimal , xs:integer , xs:boolean , xs:date , xs:time)
No support for new types	Supports creating new data types
No support for inheritance	Supports inheritance
No Support for Data Constraints and Restriction	Supports Data Constraints, Restrictions, RegEx Pattern…

For you to search !

- **Choice for Attributes using XSD ?**
- **Convert Automatically DTD to XSD ?**
- **Ignore Order of Sub-Elements within complexType ?**